Distributed Tests: An ARMY Perspective

Kenneth G. LeSueur
US Army Redstone Technical Test Center (RTTC), Redstone Arsenal, AL
ken.lesueur@us.army.mil

Ken Yetzer, Mike Stokes, Ashok Krishnamurthy, and Alan Chalker
Ohio Supercomputer Center, Columbus, OH
{kyetzer, mstokes, ashok, alanc}@osc.edu

Abstract

At the heart of Network Centric Warfare is the ability for all assets on the battlefield to communicate and coordinate their actions. Therefore, as these systems are being developed they must be tested and evaluated together along with other assets in a networked environment. The key requirement to conducting this type of Test and Evaluation (i.e., distributed testing) is having the necessary expertise to combine networking, security, high performance computing (HPC), and simulation experience as needed. The Army began preparation for testing in a distributed environment more than a decade ago when the Army Test and Evaluation Command created the Virtual Proving Ground. An outgrowth of this technology investment was a series of increasingly complex distributed test events or exercises whose purpose was to provide technology integration points and demonstrate and document the capabilities and methodologies for conducting distributed testing. The experience gained in performing these exercises over the past ten years, raises important questions regarding interoperability of network-centric assets, performance of spatially separated systems (especially those involving Hardware-in-the-Loop (HWIL) assets) and high bandwidth requirements such as video and audio streaming feeds. This paper seeks to expound on a few of these issues as observed from the most recent tests as observed from the US Army Redstone Technical Test Center (RTTC). The latest exercise, Distributed Test Event 5 (DTE-5), occurred in August/September of 2005.

1. Introduction

DTE-5 was a set of events that have different (although overlapping) participants, and objectives with more than 18 various centers networked together across the country[1]. It was a two-week effort to demonstrate joint classified distributed test and simulation capabilities. The joint portion of the exercise, called the Multi Service Distributed Event (MSDE), involved participants from the Army, Air Force, Navy, and Joint Forces Command while the Command Collaboration Effort (3CE) involved three commands within the Army, the Developmental Test Command (DTC), the Research Development & Evaluation Command (RDECOM), and the Training and Doctrine Command (TRADOC). One of the main centers used in this distributed test portion of the exercise was the RTTC in Alabama. The paper will cover three primary aspects of distributed testing: Networking, Simulations, and Analysis.

2. Networking

In the context of this paper, *distributed testing* is when two or more test assets are either spatially or functionally separated from one another. When considering performing a distributed test, the network connecting the sites is probably the most critical element within the distributed environment. The network bandwidth and latency performance immediately comes to mind but other critical performance considerations include the classification level of the test, deterministic performance of the network, encryption needs, multiple levels of security, and time synchronization.

2.1. Latency

When performing distributed testing, one critical aspect of network performance is latency. For different types of tests/simulations the latency budget can differ greatly. Latency will sometimes place limitations on what type of simulations may be run in what type of distributed environment. Distributed testing in itself has many meanings with differing levels of latency. For example, distributed testing from one laboratory to another lab in the same building may have latencies only in the hundreds of nanoseconds. A test linking multiple buildings/ranges within a single test center/installation, where you have direct fiber optic links, may have link latencies between 10–100s micro seconds. Performing cross country distributed testing over a Wide Area

HPCMP Users Group Conference (HPCMP-UGC'06)
0-7695-2797-3/06 $20.00 © 2006 IEEE

Network (WAN) such as the Defense Research and Engineering Network (DREN) will create link latencies between 10–100 milliseconds. International distributed testing can have 100s of milliseconds to several seconds of latency in the network links.

2.2. Tactical vs. Test Network

In the conduct of distributed testing it is often the case that the tactical network must be implemented over the distributed test WANS, along with the test control, instrumentation, and test monitoring functions. There exist several areas of potential problems when implementing a distributed test that the tactical layer and the test control layer share the same network infrastructure. The first is the potential for the simulated tactical network, implemented artificially over a high performance WAN, could exceed the performance (bandwidth, latency, errors/dropouts) of the actual tactical network on which the Unit Under Test (UUT) is to operate in the field. This situation would generate errant data due to the artificial high performance of the WAN. Another area of potential problems exists when fluctuations in the traffic on the test control layer of the network impacts the performance of the simulated tactical network. This situation creates a validation issue with the distributed test. To ensure that conducting the test in a distributed manner does not invalidate the test results, the network, and often the application codes themselves, must be instrumented and monitored in real time.

2.3. Classification

The classification level of the test will significantly effect the implementation of the network and all of the laboratories and test ranges that need to connect. If a single simulation or asset in the test is classified then each node on the network must be classified (with the exception of a Cross Domain Solution mentioned below) and the data must be encrypted before leaving the secure areas. Each node on the secure network must use compatible encryption equipment with matching keys. Often when different organizations are joining to a classified network there will need to be a Memorandum of Agreements (MOA) signed to facilitate the interconnection. Getting MOAs signed can be a significant schedule driver and should be addressed early in the process. The encryption devices themselves may be a long lead item that should be addressed early in the schedule.

Because of the agreements needed between organizations in joining a classified network environment, it was decided for DTE-5 that there would be three peering points on the network, one for each service (see Figure 2) and only these three locations would need MOAs. All locations within a service would have to go through the appropriate peering points when communicating with other service sites. Though making the MOA issue easier, this approach does have a downside in performance because of the extra latency in going through the peering points instead of directly connecting. A good example scenario for this would be RTTC needing to send information to China Lake. The data would route from RTTC, AL to WSMR, NM to Dahlgren, VA to China Lake, CA; essentially making three trips across the country instead of one trip if the network were fully meshed. Network topologies are being investigated to help eliminate the impact due to the peering points.

Using encryption devices in the distributed links may cause other adverse effects. During the execution of the Distributed Test Event 5 (DTE-5), KG-175 TACLANE encryptors were used to encrypt the classified data before transmission outside of each test facility. In DTE-5 it was desired to run the High Level Architecture (HLA) Run Time Infrastructure (RTI) in a multicast mode to help keep down the network traffic. The TACLANE encryptor does not support the multicast network protocol so each site in the distributed test had to distribute the data unicast and have a User Datagram Protocol (UDP) forwarder redistribute the data multicast on each local WAN.

Another network challenge encountered during the execution of DTE-5 was linking to the instrumentation installed on live systems (Manned Ground Vehicles, Unmanned Aerial Systems, Unmanned Ground Vehicles, and dismounted personnel). Because the DTE-5 network was classified, all the installed instrumentation had to operate at the same classification level. The live players were mobile during the execution of the event so their connection to the network had to be wireless. At the time the only approved encrypted wireless link available was the SecNet 11. Because of the very limited bandwidth of the SecNet 11, special considerations were given to the network design to keep the massive amount of traffic going over the WAN from overwhelming the wireless connection. Only the information needed by the live players was routed to the SecNet 11 system which kept the bandwidth within the operational bounds.

The KG-175 TACLANE encryptor used in DTE-5 is one example of an Internet Protocol (IP) based packet encryption device. In some hard real-time distributed test activities, there is a requirement to keep the encryption and transmission latency to an absolute minimum. In such cases another type of encryption and distribution method can be employed. RTTC is in the final stages of fielding a system that uses a high speed serial connection rather than the IP based network and is using a bit level encryption system rather than a packet based option such as the KG-175. Direct point to point dedicated fiber optic

lines are used to transmit the encrypted data between locations. This application is practically limited to distributed testing within a test center or installation but has big benefits in the reduction of latency and jitter within the connection.

2.4. Time Synchronization

In a distributed event, there are a variety of time scales that must be accurately recorded during the event. For example, a maneuvering vehicle such as a missile might generate 50 events/sec while a human might generate one event every five seconds. In any case, accurate representation of time is mandatory to understand the correlation of one event to another on the temporal scale. This requires that all participants in the distributed test must be calibrated to the same time base accurate to within some minimum error specification. Some early distributed tests used the Network Time Protocol (NTP) to provide clock synchronization for geographically distributed systems. The accuracy of this system is around one millisecond which is not accurate enough for some elements of the test. During the 3CE-EC2 effort, a Global Positioning System (GPS) timing method was developed by the Electronic Proving Grounds (EPG)[3] (the Tenacious Timekeeper Position Plus (T2P2)). The accuracy of the hardware was rated at one microsecond, however, it was discovered that the measured accuracy of the system was around one millisecond due to the way the operating system interacted with the hardware. The Symmetricom bc637PCI-U was also explored, but with the same results. Obviously, an inexpensive and effective solution is elusive although there are current offerings that show promise.

2.5. Cross Domain Solution

It is often the case that only a small percentage of nodes involved in a distributed test are classified and the majority of data that is shared between nodes is unclassified. This situation causes all the nodes, whether classified or unclassified, to operate at a classified level if a Cross Domain Solution (CDS) is not in place. A CDS allows the connection of a network at a higher classification to a network of lower classification (e.g., Army Test Integration Network to DREN) through a trusted bridge or guard.[2] Data that is identified as unclassified is allowed to traverse the CDS from one network to the other (see Figure 3).

There are two common situations in distributed testing that the implementation of a proper CDS could greatly aid. The first is the case where the distributed event is determined to be an unclassified event. Any node that is classified cannot participate in the event without a CDS resulting in less than ideal participation and/or fidelity in the distributed event. The second case is where the distributed test is classified. Facilities/ranges that have only an unclassified infrastructure would not be able to participate without increasing the security level of the entire range.

For certain facilities this is not practical or is cost prohibitive. With a CDS in place that can pass the unclassified traffic between the two networks, the significant cost, time, and complexity of switching security levels may be avoided. RTTC is currently in the process of certification and accreditation of a CDS that will link an unclassified network to a classified network through a HLA data bridge that operates with latency low enough to support most distributed test applications.

3. Simulation Architecture and the Test Environment

The architecture of distributed testing centers about the development and application of the networking layer used to communicate among the various distributed elements of the test environment. To understand the current status of this work, it is instructive to take a look at the various methods that have been employed for network communications, the advantages and disadvantages of these methods, and where we appear to be going.

3.1. Distributed Interactive Simulation (DIS)

The oldest and one of the most successful communication protocols is the Distributed Interactive Simulation (DIS) protocol supported by the Simulation Interoperability Standards Organization (SISO, http://www.sisostds.org). SISO was initiated in April 1989 with a small conference called "Interactive Networked Simulation for Training," that attracted nearly sixty people. The goal of the conference was to encourage interoperability and rapid advancement of standards by creating a forum for discussion of standards as networking and simulation technology matured. The group based their early standards on the work developed by the project SIMNET[4] where all the simulators were manufactured by the same vendor. The concept that extended the SIMNET program to include simulators by other manufacturers is called Distributed Interactive Simulations or DIS.

The primary network protocol for DIS is the UDP packet. The UDP packet was chosen for two primary reasons: 1) unlike the TCP protocol that creates a *connection state* between two endpoints, UDP is a *best effort* protocol. For example, when a TCP packet is sent

from a user application, an error is returned if the information on the other end was not received successfully. When a UDP packet is transmitted, an error is returned *only* if the packet was not sent successfully. The sending client is not burdened with whether or not the packet was received. This makes the UDP clients much faster than their TCP counterparts and provides the ability for humans to work with input from and output to simulators in real-time. Another advantage of using UDP is its support of multi-cast data. Large distributed simulation configurations usually group hosts on a single network to receive the same message simultaneously. This reduces the overhead of broadcasting individual messages to each computer on the network and makes possible very large distributed simulations with little additional network traffic.

The foundation of the DIS data structure is a standard set of messages and rules called Protocol Data Units (PDU). An example is the Entity State PDU that represents all of the state information of a simulated entity. For this particular PDU, the information is position, velocity, acceleration, orientation, and appearance. The objects may represent aircraft, ground or water vehicles, weapons, humans or animals, or any other entity important to the simulation. To save network bandwidth, extrapolation, or *dead reckoning*, is used for the movement of the entity if the next PDU has not arrived in time for the next update. Some of the other PDU's are Emissions (information includes point of origin, power, frequency, direction, scan pattern, and other parameters associated with electronic sensors), Bit Stream Packets (voice samples, computer-to-computer communications, images, or any other digital bit stream), Environment (atmospheric or oceanographic data), and Fire & Detonation packets (packet pairs carry the information needed to describe the firing of a ballistic [unguided] weapon, and the detonation of the projectile).

One important element in the design of simulation protocols often overlooked is that data is very short lived. Most reliability mechanisms (such as that employed by Transmission Control Protocol (TCP)) attempt to retransmit the original data to correct for packet loss. This approach might be required for conventional applications such as file transfer, but in a distributed real-time simulation, such recovery is of little use since the retransmitted data would be superseded by newer data. A better approach is a recovery mechanism that retransmits a fresh version of the data in a lost packet.

Although DIS has been around a long-time and had its success stories, it is not without faults. Lack of standardization allows messages to be constructed that no other client may recognize. Perhaps worse, there is no formal mechanism to add new message definitions to the known list. Interoperability is procedural and accomplished purely by gentleman's agreement. Since the development of DIS, new network standards have emerged that support more advanced capabilities and require adherence to specifications. Two of the most common are the High-Level Architecture (HLA) and the Test and Training Enabling Network (TENA).

3.2. HLA and TENA

HLA and TENA are both standards developed by the Department of Defense (DoD) to achieve interoperability and reuse within the defense community. Both are built atop the Common Object Request Broker Architecture (CORBA) developed by the Object Management Group (OMG, http://www/omg.org). To understand the steps in developing a HLA or TENA application, you should consider the COM (Component Object Model) used by a CORBA application. A CORBA object is defined using the Interface Definition Language (IDL) similar to a C++ object. A utility is run that reads the IDL files and generates stub or skeleton code in your favorite computer language (FORTRAN, C, C++, JAVA, and others are supported). The user then writes the methods defined by the objects based on a standard naming convention that defines the actual data and methods functions promised in the IDL file. Standard calls are available within a CORBA environment that provides naming services for identifying an object by name across the network. Other calls generate and manage object repositories, and many other services. The standard way of developing applications is to write one program (an object factory) that creates objects with a main program that does nothing but waits on other programs to remotely invoke methods on its functions. Once the object factory is running, one or more client applications are created using the same namespace as that used by the object factory application, and requests a pointer (using that wonderful namespace service) to a remote object upon which methods can be called by the local application. There are a number of salient features of this software design: 1) Objects created on one computer can have its methods invoked from another computer (the intended design). This works well if the object is large and the amount of data movement within the object is small compared to its size, 2) Object factory applications and the client applications do not have to be written using the same computer language. The object factory could be written in C++, for example, while one client could be written in JAVA, and yet another in C, and 3) Object definition is independent of its implementation. As long as the object interface definition (i.e., the argument list) remains fixed, the user has little interests in knowing the details of how the task is accomplished. This is true for all object models, but is mentioned in the context of simple protocols such as DIS that have no notion of object methods.

While CORBA and its derivative protocols is a well-defined standard and is being used to great success, it has its faults. Developing a CORBA application is not a trivial task. Through there are standard libraries and utilities to support CORBA such as J2EE, it is a complex multi-step process that requires a large monetary and time commitment on the part of developers. Secondly, CORBA is considered by most to be a heavyweight application with many layers of software for data marshaling, conversion, and transport. Developed primarily for business applications, it is not clear that CORBA applications are the optimal choice for real-time scientific applications. One must wonder if the process really needs to be this complicated and impose so strong a real-time penalty for its operation?

While HLA and TENA share common technology, they were not designed with the same focus. HLA was designed as a method for communications between simulations, while TENA was designed for communications between instrumentation. Each perform well in their respective domains, but in a distributed test with interaction between instrumentation and simulations, neither protocol produces an ideal solution, particularly in terms of scalability to millions of entities as will be needed for future scenarios. The current short-term solution is to provide for a mixed architecture where TENA and HLA components communicate through gateways (see following section). A longer-term solution is to develop a new standard protocol that satisfies the demands for interoperability, software reuse, support for both instrumentation and simulation, and scalability for large (millions) scale distributed tests. The big question is whether the T&E community is ready to concede that there is much work still left in protocol development, and if so, which organization will take the lead in developing new standards?

3.3. Mixed Architectures

DTE-5 was a mixed architecture environment that included DIS, TENA, HLA using the Modeling Architecture for Technology, Research, and Experimentation (MATREX) Federation Object Model (FOM), and HLA using the DTE FOM (see Figure 4). DTE-5 allowed all participants to bring new or legacy simulations, hardware, and instrumentation in an *as is* configuration, leaving the majority of the integration/translation to the architecture to solve. To integrate these architectures into a single functional capability, the participants at each site had to agree to some design constraints and run codes locally to perform the link between the architectures. All *long haul*, or site to site, connections were performed using the HLA. Locations using DIS were required to run a HLA/DIS gateway and a UDP forwarder to distribute the messages on the Local Area Network (LAN). A custom designed *federation-to-federation bridge* was executed at RTTC to exchange traffic between the MATREX and DTE FOMs. This approach is a challenge on the architecture but allows participants to operate in the distributed exercise using legacy simulations/codes that may be under configuration control or passed the Verification, Validation, and Accreditation (VV&A) milestone that would limit the changing of the simulation.

3.4. Shared Memory Multi-Architecture (SMMA) Interface

There is an old joke that goes something like “the troubles with standards are that there are so many to choose from”. The need to support legacy hardware and software developed using older standards (or various versions of a single standard) coupled with the need to support newer formats can be a daunting task. To provide a partial solution to this problem, RTTC has developed a Shared Memory Application Programmers Interface (API) that allows multiple architectures to co-exist simultaneously while also supporting new and evolving models in a single network environment. The design goals for the API were: API must be easily implemented into the existing code base with minimal training required; New and existing object models should be easy to implement; Must have little or no impact on the performance of hardware and software running in real-time; Changes to the API must be infrequent; and Develop applications without protocol specific source code.

The operational philosophy behind the SMMA interface is simple; consider that multiple federates (applications) are running on a shared memory computer and wish to participate in a distributed test. Each federate places a DIS message, HLA or TENA object, or any other recognized data format, into shared memory segments. A server process also running on this computer recognizes each data format and acts as an ambassador between the objects in shared memory segments and the other components of the distributed test. Whenever each application wants to publish its data component, the application simply notifies the server process. The server process in turn takes care of formatting the data item in any of the other known protocols and shuttles it over the network as needed. This architecture requires that only one central application have knowledge of all the various formats. For every DIS message, for example, there would be upon demand, an equivalent HLA and TENA object representation. Furthermore, the process is bidirectional. Whenever a remote message or object has been published, the necessary shared memory segments are created and its equivalent format stored in the

segment. The local application is then informed that an incoming message/object is available.

3.5. eXtensible Modeling and Simulation Framework (XMSF)

The objective of this effort[5] is to develop, promote, and establish standards, specifications, and practices for web-based technologies in military Modeling and Simulation systems; Plan and conduct workshops and conferences for promotion and study of XMSF concepts, and develop and demonstrate exemplar programs to illustrate application of XMSF principles for improved system interoperability. The premise of the concept is that "... the only software systems that composably scale to worldwide scope utilize World Wide Web technologies." While it is questionable that real-time features can be handled solely with web technology, it is no doubt that browser technology will play a role in monitoring and perhaps steering elements of distributed testing. Certainly this is a technology that should be monitored as it matures for possible adoption by the T&E community.

3.6. Tactical vs. Ground Truth Data

When executing a Live, Virtual, Constructive (LVC) distributed test, often with the tactical network operating on the test network infrastructure, it is imperative to ensure the proper separation of tactical or Unit Under Test (UUT) data from ground truth data. Ground truth data is the data collected from calibrated instrumentation external to the UUT. It is sometimes easy to let the two types of data get inadvertently linked causing the test to be invalid. This is especially true with voice communication links and common operating pictures. Usually the ground truth data is more accurate than the same tactically generated data. The errors in the tactical data must be allowed to propagate through the system to properly assess the performance. All data generated in the event should be labeled to prevent such occurrences. Test conductors and personnel playing tactical roles should never share the same physical room during the event so that the tactical operator will not be contaminated with information not otherwise known if it were a pure tactical engagement.

3.7. Challenges of Live, Virtual, and Constructive (LVC) Test Environment

The integration of Live, Virtual, and Constructive assets in a test environment bring many challenges to the test conductor, simulation Subject Matter Expert (SME), and the analyst. One such challenge encountered in DTE-5 was the need to start the exercise with the Semi Automated Force (SAF) representing three Combined Arms Battalions (CAB) 10s kilometers from the first objective where a single live platoon was to dismount and engage live red forces. The test range that the live players were to execute their dismount mission on was a very small area and the vehicles could not actually travel the long distance from the virtual start point to the actual range. The solution was to have the live platoon be represented by a separate copy of OneSAF during the travel from the start location to the live play box. A start location for each live vehicle was surveyed and marked on the range. The SAF would end its initial mission for this platoon at the surveyed points. At a predefined time in the scenario, the SAF representing the live platoon was turned off and the instrumentation packages on the live vehicles were enabled and started publishing the information for the platoon. After the live mission was over, a similar transition was performed so this platoon could interact with the other constructive forces in engagements later in the scenario. This transfer from constructive to live, and back to constructive worked well for the heavily scripted scenarios used in DTE-5. In test events that allow more *free play* this approach may not be optimal.

4. Analysis

Clearly the design of the most recent DTE/MSDE/3CE distributed events has been proof-of-principle tests, or has often been quoted by the principal author as being a "test of a test". These steps are necessary to develop the confidence that the underlining technical and cultural barriers may be overcome. While the current tests have produced a large amount of data in many different media such as video, audio, digital, and others, little analysis to date has been performed on the data beyond studying the performance of individual systems. As the technology of distributed testing matures, the architecture of the tests will have to change to address specific questions originating from new weapon design, and acquisition and tactical considerations regarding the effective use of new weaponry. The PET component of the High Performance Computing and Modernization Program is evaluating methods aimed at data mining the massive amount of data resulting from a distributed test. But before data can be mined, what type of problems might we be asked to solve?

Questions that might arise are: 1) how will a potential new weapon system perform based on the performance of an existing weapon in the inventory?; 2) is the life-cycle cost of standing up a new weapon system justified if its performance over existing weapons is small compared to its costs?; 3) how well will a new weapon system inter-operate with existing systems?; and 4) what

level of education and training will be required for future war fighters to operate new weapon systems. To answer these questions, higher fidelity virtual and constructive simulations must be designed to interoperate with HWIL and Human-in-the-Loop (HITL) elements. More importantly, the results of these tests must be data-mined not as a set of individual systems, but as a single data-fused system. This is particularly important with HITL applications where both audio and video data review would be needed, for example, to determine target identification time.

One of the major challenges facing the T&E community with conducting distributed LVC testing is the VV&A process. For tests on systems that will be formally evaluated, VV&A of all the participating simulations, networks, instrumentation, and data collection sites must be completed. Each live asset or simulation that has a dependence on data coming from an outside source must verify that the information is validated. Often in distributed testing one must depend on data and V&V information from outside (non local) commands or services.

5. Conclusions

DTE-5 was very successful both in terms of testing various aspects of LVC distributed testing in a classified environment and showing in general how distributed testing may be done properly. A lot of knowledge was gained from this exercise that will be used for future testing of FCS systems and by teaching others (e.g., other branches of the military) how to conduct their own distributed tests with PET involvement. The IMT PET team has compiled observations and insights from the test into a half-day seminar that will be given to test engineers at other test centers. The intent of the seminar is to disseminate 'lessons learned' in order to facilitate future distributed test events.

Distributed testing is crucial to the DoD's vision of Network Centric Warfare by all the branches of the military. Therefore, it is vital that successful distributed test events be documented in such a way that it can be taught to others who require this type of testing.

Acknowledgements

This publication was made partially possible through support provided by DoD HPCMP PET activities through Mississippi State University under contract. The opinions expressed herein are those of the author(s) and do not necessarily reflect the views of the DoD or Mississippi State University.

References

1. Barton, et al, "Experience with Distributed Testing." *ITEA Journal*, Mar/Apr 2006.

2. Welke, Steve, "Navigating the SABI/CDS Process." *Trusted Computer Solutions*, Sep 2005.

3. Hynes, Mark, "Electronic Proving Ground (EPG) Technology Snippets." *ITEA Journal*, Dec/Jan 2004.

4. Hofer, R. and M. Loper, "*DIS today*." *Proceedings of the IEEE, Special Issue on Distributed Interactive Simulation*, Vol. 83, No. 8, pp. 1124–1137, August 1995.

5. Don Brutzman, https://www.movesinstitute.org/xmsf/people.html.

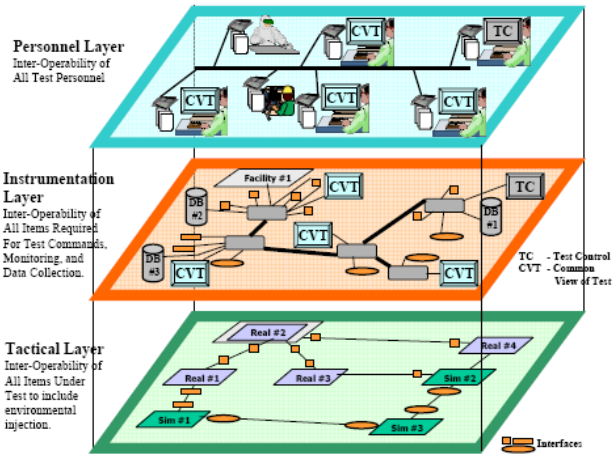

Figure 1. Network layer is distributed testing

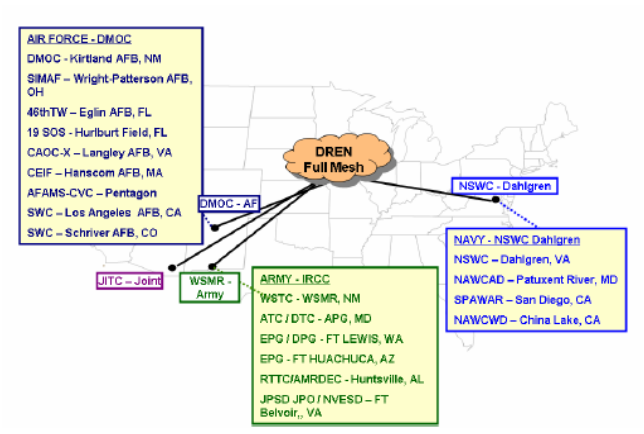

Figure 2. DTE-5 network peering points and participating centers

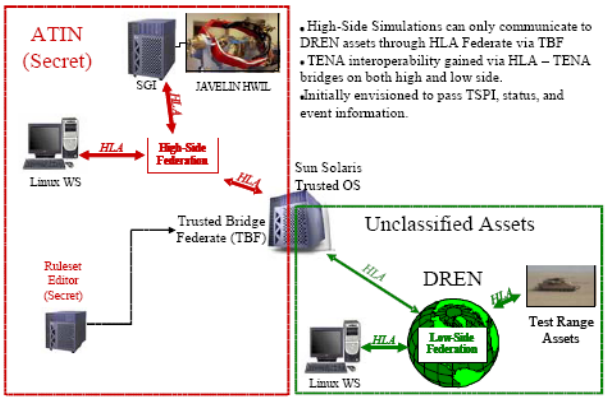

Figure 3. Cross-domain solutions

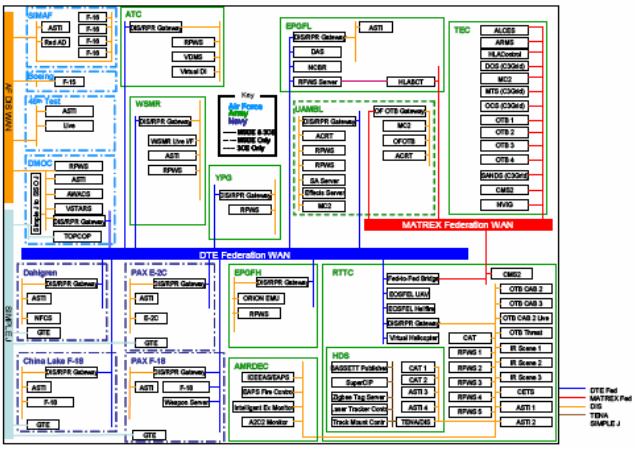

Figure 4. DTE-5 architecture